



# Best Practices for Exchange Database Management White Paper

---

## Abstract

This whitepaper discusses how Microsoft® Exchange maintains database consistency and helps users understand how the system works so they can make more informed policy decisions regarding the maintenance of their Exchange databases.

**Note:** This paper was written for Exchange Server 5.5 Service Pack 1 which has been replaced by Exchange 5.5 Service Pack 4 (SP4), the most recent service pack. SP4 is a cumulative release and includes all changes for SP1. See [Exchange Server 5.5 Service Pack 4 Overview](#) for more information about SP4.

*The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.*

*This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.*

*Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.*

*Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.*

*Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, email address, logo, person, place or event is intended or should be inferred.*

*© 1998 All rights reserved.*

*Microsoft, the BackOffice logo, Outlook, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.*

*The names of actual companies and products mentioned herein may be the trademarks of their respective owners.*

---

## CONTENTS

INTRODUCTION .....	2
EXCHANGE DATABASE ARCHITECTURE .....	3
Information Store	3
Database Components	3
MAINTAINING DATABASE INTEGRITY .....	6
Protecting Data Using Log Files	7
Tracking the Starting Point for Recovery	9
Circular Logging — Don't Use It!	11
TYPES OF FAILURES .....	11
Physical Corruption	11
Preventing Physical Corruptions	12
Repairing Physical Corruptions	12
Logical Corruption	13
Preventing Logical Corruptions	13
Repairing Logical Corruptions	14
BACKING UP DATA .....	15
Online Backup	15
Ensuring That All Data Is Backed Up	15
Offline BackUp	16
Backup Strategies	16
Maintenance Routine	16
RECOVERING DATA .....	18
Soft Recovery	18
Restoring from Online Backup	18
MANAGING AND MAINTAINING YOUR DATABASES.....	20
Defragmenting Your Databases	20
Capacity Management	20
PLANNING HARDWARE NEEDS .....	22
Optimizing Database Access	22
Optimizing Backup and Restore Performance	22
KEY MESSAGES .....	24

---

---

## INTRODUCTION

For many companies today, the ability to communicate quickly and effectively using e-mail is vital to staying on top of the competition. Companies can grind to a halt when e-mail is down because it's such an important part of communicating with people both inside and outside of the company. As a result, e-mail administrators have a very important job—they're responsible for keeping e-mail up and running at all times with minimal interruptions. One of their biggest concerns is database integrity and ensuring that information exchanged between users isn't lost.

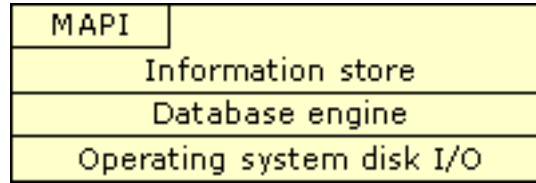
This white paper discusses how Microsoft Exchange maintains database consistency and helps users understand how the system works so they can make more informed policy decisions regarding the maintenance of their Exchange databases.

---

## EXCHANGE DATABASE ARCHITECTURE

To understand how Exchange keeps track of information in the system, it's important to first understand how the Exchange database structure is put together.

The following diagram shows how the Exchange architects think of the various database components in the system.



*Figure 1: Database Component Architecture*

### Information Store

The information store is key to database management in Exchange. This component is responsible for storing data, such as e-mail messages in user mailboxes and information in public folders. The information store is actually two separate databases. The private information store database, called Priv.edb, manages data in user mailboxes. The public information store, called Pub.edb, manages data in public folders.

The information store works together with the Messaging Application Programming Interface (MAPI) and the Exchange database engine to ensure that all user actions are recorded on the server's hard disk. For example, when a user saves a message in the Microsoft Outlook® messaging and collaboration client, MAPI first calls the information store, which then calls the database engine, which then writes the changes to disk.

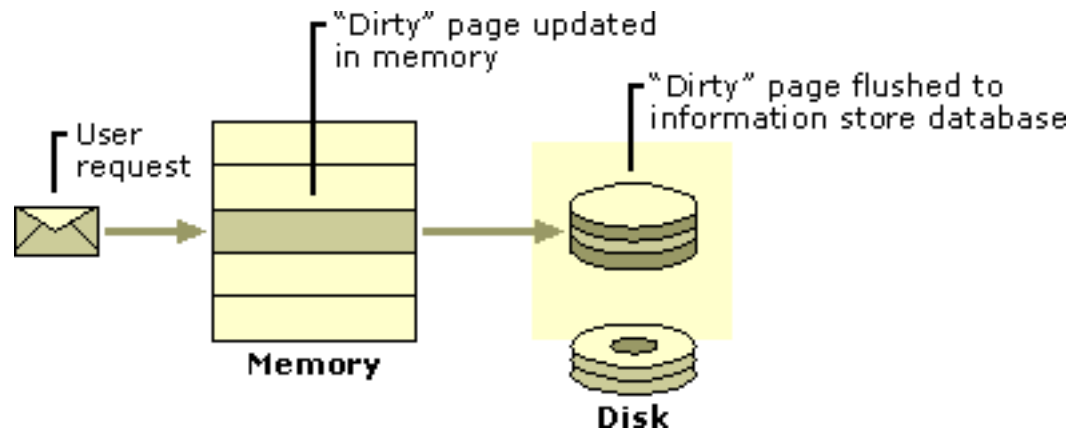
### Database Components

The design of Exchange is based on standard database technology. The system relies on an embedded database engine that lays out the structure of the disk for Exchange and manages memory. The database engine technology used in Exchange is also used "under the covers" by other applications for the Windows® operating system, such as WINS and DHCP. And, Microsoft's SQL Server™ database product uses its own kind of database engine that's based on similar technology.

The Exchange database engine caches the disk in memory by swapping 4 KB chunks of data, called pages, in and out of memory. It updates the pages in memory and takes care of writing new or updated pages back to the disk. This means that when requests come into the system, the database engine can buffer data in memory so it doesn't have to constantly go to disk. This makes the system more efficient because writing to memory is "cheaper" (or faster) than writing to disk. When users make requests, the database engine starts loading the requests into

---

memory and marks the pages as “dirty” (a dirty page is a page in memory that has been written with data). These dirty pages are then later written to the information store databases on disk.



*Figure 2: Writing Data to Disk*

Although caching data in memory is the fastest and most efficient way to process data, it means that while Exchange is running, the information on disk is never completely up-to-date. The latest version of the database is in memory, and since many changes in memory haven't made it onto disk yet, the database and memory are out of sync. If there are any dirty pages in memory that haven't been flushed and written to disk yet, the databases are flagged as inconsistent. So, as strange as it sounds, as soon as Exchange starts, while Exchange is running normally, the databases are technically inconsistent. The only time that Exchange databases are truly in a consistent state is when all the dirty pages in memory are flushed to disk following a graceful shutdown in which no errors occurred.

Now, it doesn't take long to realize that if something goes wrong, it is difficult to recover. If you lose the contents of memory (say, the server crashes) before the data is written to disk, you're left with an inconsistent database. As a result, Exchange needs a way to recover from this situation. In other words, Exchange needs a way to secure a copy of everything that has changed in memory until it's written to the database on disk. And, this is the role that transaction log files take on. There's more on that in the next section.

---

### **How Exchange Dynamically Changes Its Buffer Size**

Before Exchange Server version 5.5, the buffer cache that the database engine used was of a fixed size. If Exchange was thrashing and needed more memory, the administrator had to manually change the buffer size. With Exchange 5.5, however, the database engine team made some innovative changes to the way that the database engine uses memory. They built dynamic buffer allocation into the

---

database engine, so that the database engine dynamically grows or shrinks depending on how much memory is available and whether there is back pressure from other services running on the Microsoft Windows NT® Server-based computer. If memory is not being used by other services in the system, such as Microsoft Internet Information Service (IIS), the Exchange database engine will take up as much memory as it needs. When other services need memory, the Exchange database engine will give up some of its memory by flushing pages to disk and shrinking the size of its buffer.

---

## MAINTAINING DATABASE INTEGRITY

People talk about Exchange as a “transaction-based” e-mail system and the information store as a transactional database. So, what *does* that mean anyway?

A transaction is defined as a set of changes, such as inserts, deletes, and updates, to a database in which the system obeys the following invariants (ACID):

### **Atomic**

Either all the operations occur or none of them occur.

### **Consistent**

The database is transformed from one correct state to another.

### **Isolated**

Changes are not visible until they are committed.

### **Durable**

Committed transactions are preserved in the database even if the system crashes.

The database engine “commits” a transaction only when it can guarantee that the data is “durable” or “persistent,” and protected from crashes or other failures. The database engine will only successfully commit data when it’s sure it has flushed that data from memory to the transaction log file on disk.

For example, to move an e-mail message from one folder, such as “Inbox,” to another folder called “Important,” Exchange must perform the following operations.

1. Delete the e-mail message from “Inbox.”
2. Insert the e-mail message into “Important.”
3. Update the information about each folder to correctly reflect the number of items in each folder and the number of unread items.

Because these operations are done in a transaction, Exchange will perform none or all of these operations. As a result, it doesn’t matter which order Exchange performs the operations. The message can be deleted safely from “Inbox” first because the system knows that the delete will only be committed if the message is also inserted into “Important.” Thanks to transactions, even if the system crashes, it is guaranteed that Exchange will never lose an e-mail message while moving it. What’s more, Exchange will never end up with two copies of an e-mail message that was moved.

The following illustrates the process of “normal logging” where data is written to transaction log files. This is described in detail in the following sections.

1. The user sends a message.
2. MAPI calls the information store to tell it that the user is sending the message.
3. The information store starts a transaction in the database engine and makes the corresponding changes to the data.

- 
4. The database engine records the transaction in memory by dirtying a new page in memory.
  5. At about the same time, the database engine secures the transaction in the transaction log file and creates a log record. When the database engine reaches the end of a transaction log file, it rolls over and creates a new log file in sequence.
  6. The database engine writes the dirty page to the database file on disk.
  7. The checkpoint file gets updated.

### Protecting Data Using Log Files

Most people naturally think that the database files are the most important aspect of data recovery. But, transaction log files are actually more important because they reflect what will happen with the data, not what has happened. In fact, the transaction log files contain information that the database files don't. As a result, you should guard log files with your life, and put them on high performance disks, even if that means putting the database files on slower disks.

Exchange uses transaction log files to keep track of the information in memory that hasn't yet made its way to the database on disk. Transaction log files are a sequence of files whose purpose is to keep a secure copy on disk of volatile data in memory, so the system can recover in the event of a failure. If your system crashes and the database is undamaged, as long as you have the log files, you can recover data up to the last committed transaction before the failure. To be most useful, it's best if the log files are on a dedicated disk, so the logs aren't affected by any disk failures that corrupt the database.

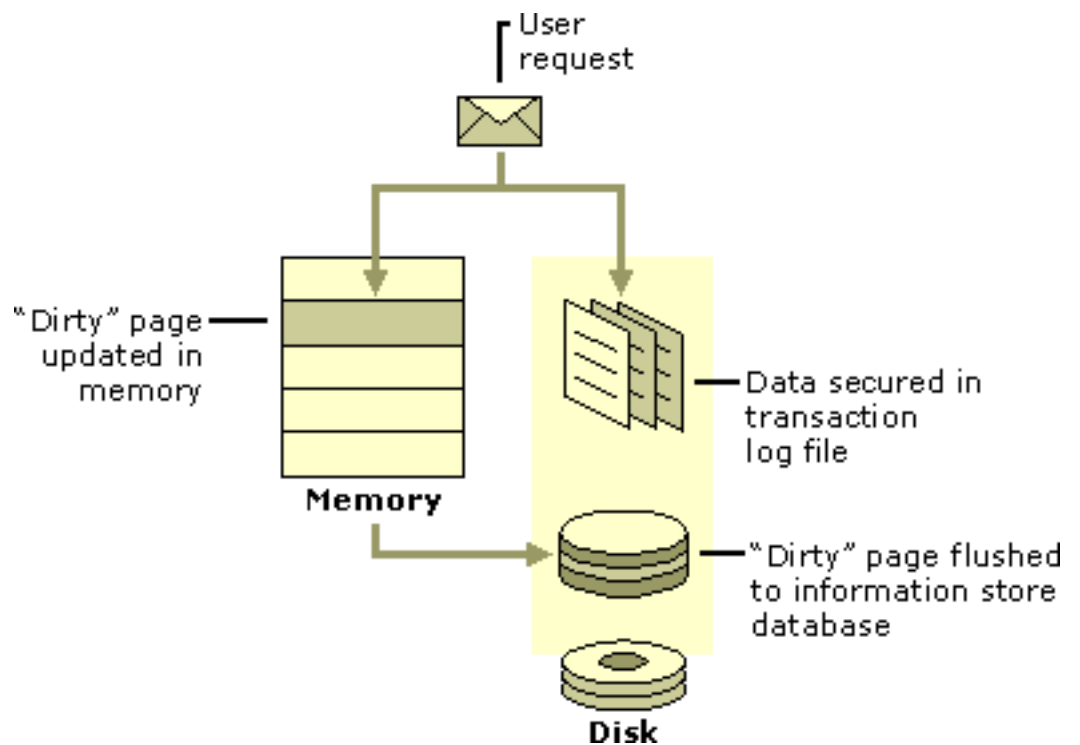
In addition to providing a safety net for data in memory, transaction log files also make writing data fast, since updating pages sequentially in a log file is a cheaper operation than figuring out the correct place to insert pages in the database. When a change is made to the database, the database engine updates the data in memory and synchronously writes a record of the transaction to the log file that tells it how it could redo the transaction in case the system fails. Then, the database engine writes the data to the database on disk. To save disk I/O, the database engine doesn't flush pages to disk on every transaction, but instead saves them up in batches.

Each log file in a sequence can contain up to 5 MB of data. When the log file is full, it rolls over to a new log file. To keep track of the log files, Exchange associates each log file with a generation number. For example, when Exchange starts for the first time, it creates a log file called Edb.log with a generation number of 1. When that log file is full and Exchange rolls over to a new log file, the new log file becomes Edb.log with a generation number of 2, and the old Edb.log file is renamed Edbnnnn.log. This continues until the next sequence of log files starts. If you were to shut down the server and lose all the log files, when you restart the information store, Exchange will create a new sequence of log files starting with a

generation number of 1. Since log files can have the same name, the database engine stamps the header in each file in the sequence with a unique signature so it can distinguish between different generations of log files.

Logically you can think of the data as moving from memory to the log file to the database on disk, but what actually happens is that data moves from memory to the database on disk. The log files are optimized for high-speed writes, so during normal operations, the database engine never actually reads the log files. It only reads from the log files if the information store service stopped abnormally or crashed and the database engine needs to recover from the failure by replaying the log files.

The following illustration shows how Exchange logs data.



*Figure 3: Logging Data*

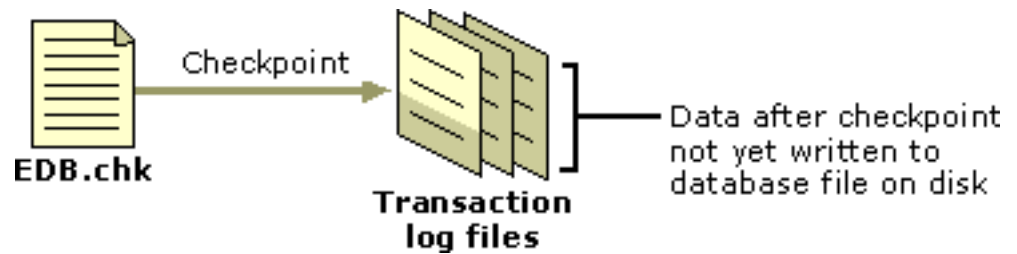
#### Tracking the Starting Point for Recovery

To keep track of the data that hasn't yet been written to the database file on disk, the database engine maintains a checkpoint file called Edb.chk for every log file sequence. The checkpoint file is a pointer in the log sequence that maintains the status between memory and the database file on disk. It indicates the point in the log file where the information store needs to start the recovery from if there's been a failure. In fact, the checkpoint file is essential for efficient recovery because if it didn't exist, the information store would have to attempt recovery by starting from

the beginning of the oldest log file it found on disk and then check every page in every log file to determine whether it had already been written to the database. This process, of course, would be very time consuming, especially if all you wanted to do is make the database consistent.

The checkpoint file is located on the system disk, so, if you have to recover your system disk, it's likely that this file is missing or you have an invalid version. But, in most cases, the checkpoint file takes care of itself.

The following illustration shows the data in transaction files after the checkpoint that has not yet been written to disk.



*Figure 4: Tracking the Checkpoint*

#### Circular Logging—Don't Use It!

In addition to normal logging which protects data until it is written to disk, Exchange supports a feature called circular logging. The circular logging feature is a hold-over from the days when server disk space was at a premium. Back in the early days of Exchange, many customers were more concerned about running out of disk space than they were about recovering data. As a result, to maintain a fixed size for log files and prevent the build up of log files, circular logging is on by default. This means that once the log file's 5 MB limit is reached, the database engine deletes it and creates a new log file in the sequence. By doing this, Exchange keeps only enough data on disk to make the database consistent in the event of a crash.

In most cases, if you're concerned about protecting your data, the first thing you should do is turn circular logging off on your computer running Exchange. While circular logging reduces the need for disk space, it eliminates your ability to recover all changes since your last backup if your information store is corrupted due to a hardware failure. This is because circular logging only maintains a small number of log files to ensure transactional integrity when recovering from non-hardware failures. Circular logging negates the advantages of using a transaction-based system and sacrifices the ability to recover if something goes wrong with the system and the contents of memory are lost. As a result, it's not recommended for most Exchange installations. To guarantee recovery from any information store corruption, you should not use circular logging.

Circular logging works in much the same way as normal logging except that the checkpoint file is no longer just an optimization feature, it's essential for keeping track of the information that has been flushed to disk. During circular logging, as the checkpoint

---

file advances to the next log file, old files are reused in the next generation. When this happens, you lose the ability to use the log files on disk in conjunction with your backup media to restore to the most recent committed transaction.

If you're concerned about log files consuming your disk resources, there's a better way to clean up your log files—simply perform regular online backups. Backup automatically removes transaction log files when they're no longer needed.

---

## TYPES OF FAILURES

Every once in a while the system fails and Exchange experiences a failure that requires the system to attempt to get back to a consistent state. Most of the time, the failures are caused by hardware problems. For example, in the last eight months, Microsoft's internal IT department has had only three failures (across 80 servers running Exchange version 5.5) that required them to restore from tape backup—all three problems were due to bad hardware.

It's important to understand that there are different types of database corruptions with varying symptoms. As a result, you can't fix all database corruptions in the same way. You need to use different tools and techniques to diagnose and fix problems.

The following are the most common types of corruptions.

### **Physical corruption**

At the lowest level, your data can become physically corrupt on the disk. This is usually a hardware related problem and always requires you to restore from backup.

### **Logical corruption**

The most typical logical corruptions occur at the database level. For example, database engine failure may cause index entries in the database to point to missing values. Logical corruptions can also happen at the application level, for example, in mailboxes, messages, folders, and attachments. An application level corruption might cause incorrect reference counts, incorrect ACLs, a message header without a message body, and so on.

## Physical Corruption

Probably the worst problem you can run into is a physical corruption that destroys your data. Physical corruptions to the database are serious because you can't repair them. To get your system up and running, all you can do is restore from backup. As a result, it's very important that you detect physical corruptions early and resolve the problems quickly.

Although Exchange typically displays a -1018 or -1022 error in the event log when there's a physical corruption, you can be proactive about detecting physical corruptions by performing online backups. Besides being the recommended method for backing up data, online backup is also the best way to determine whether there's a corruption in a database file since it's the only process that systematically checks every single page in the database. For example, when you run online backup, the Backup software reads each 4 KB page in the database file, passes it to the database engine, and then writes the pages to tape. Each time the database engine accesses each page, it verifies that the checksum on the page is correct. If the checksum on the page doesn't match the checksum that the database engine calculates, there's a physical corruption in your database on disk and Backup will end with a -1018 error.

---

## Errors That Typically Indicate Physical Corruptions

If you see the following errors in your event log, most likely you have a physical corruption in the information store.

### -1018 (JET\_errReadVerifyFailure)

The data read from disk is not the same as the data that was written to disk.

### -1022 (JET\_errDiskIO)

The hardware, device driver, or operating system is returning errors.

### -510 JET\_errLogWriteFail

The log files are out of disk space or there is a hardware failure with the log file disk.

## Preventing Physical Corruptions

The best way to prevent physical corruptions is to outfit your server with quality hardware components and configure your system correctly. You should also make sure you're not running file-level utilities (such as anti-virus software) against database and log files on your computer running Exchange. The good news is that if you have reliable hardware, you will probably never see a physical corruption. But, if you consistently run into -1018 errors, most likely you have a bad disk or disk controller, or some other hardware problem that you need to resolve.

---

## Should You Use Write-back Caching or Not?

Some write-back caching array controllers incorrectly return successful commits on transactions before the data has been secured to disk. Your safest course of action is to disable write-back caching, unless you have a battery-backed up write-back caching controller that writes information to disk. If you do use write-back caching, make sure that data is fully protected and that you have procedures in place to make sure data in a cache is always replayed to the right disks after a crash. This is important because if you lose data in the cache, you'll most likely end up with a corrupted database.

## Repairing Physical Corruptions

There is really no safe way to repair a physical corruption to the database. You can run Eseutil.exe in repair mode to get the database functioning again. However, this is not recommended since Eseutil simply deletes the bad page and you end up losing data.

The **only** way to recover from a physical corruption in your database is to restore from your last good backup (if your previous backup ran without errors, you know it's good) and roll your log files forward to bring your system up to a consistent and non-corrupt state. If your backup fails again, most likely you have a problem with the

---

disk where the database is located.

---

### Avoid Using Eseutil /p if Possible

Eseutil is a tool that ships with Exchange that can be used as a last resort, when all else fails to repair database corruptions. Eseutil in repair mode (**Eseutil /p**) works at the database engine level to get a broken database up and running again by simply deleting corrupted pages. As a result, Eseutil should never be used to recover data.

After running **Eseutil /p**, you must run **Isinteg –test alltests –fix** to restore a consistent state.

### Logical Corruption

A logical corruption is much more difficult to diagnose and fix than physical corruptions because they are unpredictable and typically caused by software bugs. Sometimes you won't even know there is a logical corruption unless something happens that alerts you to a problem.

**Note:** Logical corruptions are extremely rare in Exchange 5.5.

If you encounter a logical corruption, you need to build your troubleshooting skills as quickly as possible. This doesn't mean you need to be the world's greatest expert on Exchange, but it does mean that you need to work consistently and methodically, and write down everything that you do.

### Preventing Logical Corruptions

Because logical corruptions are so unpredictable in nature, there is no foolproof method you can use to prevent them. However, here are some tips to follow to reduce your risk.

- Install Microsoft Exchange Server version 5.5 Service Pack 1, the latest service pack for Exchange, as soon as possible. It fixes a number of known problems in Exchange 5.5. For a complete list of the specific fixes, see Knowledge Base article Q171593, "List of Bugs Fixed in Exchange Server 5.5 Service Pack 1."
- Make sure that no one is changing your Exchange configuration without your knowledge.

If you've done all of that and you still have a problem, then you've discovered a new bug. You should notify Microsoft as soon as possible so we can resolve the issue in the next product update.

---

## Repairing Logical Corruptions

Logical corruptions can occur in the information store or the database engine. Because logical corruptions can seriously damage your data, you shouldn't ignore errors in the information store or the database engine. You can use ISINTEG to check problems with the information store, or ESEUTIL to check problems with the database engine.

**Important:** You should use ISINTEG and ESEUTIL as a last resort only after you've tried to restore your system from backup.

### ISINTEG

The Information Store Integrity Checker (ISINTEG) finds and eliminates errors from the public and private information store databases. These errors can prevent the information store from starting or prevent users from logging on and receiving, opening, or deleting e-mail. For example, you can use ISINTEG to correct information store counters in memory when they get out of sync following a system crash. ISINTEG is not intended for use as a part of normal information store maintenance. It is provided to assist in disaster recovery situations where the database has become damaged.

Because ISINTEG works at the logical schema level, it can recover data that ESEUTIL can't. This is because data that is valid for ESEUTIL at the physical schema level can be semantically invalid at the logical schema level. ISINTEG logs information to the event log so you can track the progress of the recovery.

ISINTEG performs two main functions:

- It can patch the information store after a restore from an offline backup
- It can test, and optionally, fix errors in the information store.

For more information about ISINTEG, see the `Isinteg.rtf` document on the Exchange 5.5 compact disc in the `Support\utils` directory.

### ESEUTIL

ESEUTIL defragments, repairs, and checks the integrity of the information store and directory. Unlike ISINTEG, which is sensitive to the use and content of data in the information store, ESEUTIL examines the structure of the database tables and records. ESEUTIL in repair mode simply deletes corrupt pages in order to get the system up and running again. Therefore, if you're trying to recover data, you should run ESEUTIL only after you've tried to restore from backup.

For more information about ESEUTIL, see the `Eseutil.rtf` document on the Exchange 5.5 compact disc in the `Support\utils` directory.

---

## BACKING UP DATA

Now that you understand that Exchange is a transaction-based system, it should be no surprise to you that it's a bad idea to perform a file-level (also called offline backup) of the database files on disk. The best way to ensure that you're preserving all data in the system, including transactions that have not yet been flushed to disk, is to perform regular online backups.

### Online Backup

Online backup enables you to back up Exchange databases to your backup medium without shutting down the server. When Exchange is performing an online backup, all services, including the information store, continue to run normally. For example, pages continue to be updated in memory and flushed to the database files on disk, transactions are recorded in the log files, the checkpoint file continues to move along, and so on.

When performing online backups, you should regularly check the event log to make sure that your online backups are completing successfully.

### Ensuring That All Data Is Backed Up

As mentioned earlier, pages continue to be updated in memory and on disk even while the backup software is backing up database files to your tape device. To make sure that all the transactions that occur after a certain point are backed up, Exchange uses a .Pat file that keeps track of the pages that are updated while the backup software is running. Without the pat file, pages that are modified during Backup won't be backed up. There are actually two .Pat files—Priv.pat is the patch file for the private information store, and Pub.pat is the patch file for the public information store.

### The Process of Online Backup

The backup software does the following during a full or copy backup.

1. Backup copies the database to the tape.
2. A subset of pages that changed after being copied to tape are added to the .Pat file.
3. The current Edb.log file is renamed to Edbnnnnn.log and a new log generation is created.
4. If this is a full backup, the .Pat file and all log files after the checkpoint (except the new Edb.log) are backed up onto the tape. If this is a copy backup, all log files before and after the checkpoint are backed up.
5. If this is a full backup, transaction log files older than the checkpoint are deleted. If this is a copy backup, no transaction log files are deleted.

The backup software does the following during an incremental or differential backup.

1. The current Edb.log file is renamed to Edbnnnnn.log and a new log generation is created.
2. Backup copies all log files (except the new Edb.log file) to the tape.

- 
3. If this is an incremental backup, transaction log files older than the checkpoint are deleted. If this is a differential backup, no log files are deleted.

## Offline Backup

As mentioned earlier, offline backups are a bad idea. Offline backup is a much more manual process than online backup. Besides being a more labor-intensive process than online backup, if you don't know what you're doing, you can really mess up your system. When you do online backup, the backup software manages files for you, but when you do an offline backup, you need to do that work yourself. What's more, when you do offline backups, you don't have a process for validating the checksum on each page of the database. By not using online backup, you throw away the single most valuable tool for detecting corruptions and performing database integrity checking.

## Backup Strategies

It's a good idea to think about your backup and restore strategy before something goes wrong. You'll be glad you did the next time you're under pressure to restore your system after a crash.

When designing a backup strategy, first think about what you want the restore experience to be like. Your procedures should be as simple as possible, with the fewest number of steps, so you can easily perform the same steps when you restore on every server.

The best strategy is to perform full online backups every day. Some people prefer to do a full backup once a week with differential backups for the rest of the week. However, if your server crashes in the middle of the week and you have to go back to a full backup done in a previous week, but you can't restore your latest full online backup (for example, because there is a problem with the backup tape), then you won't be able to play back all the transactions up to the time of the crash.

## Maintenance Routine

You should implement a maintenance routine that enhances your chances of successfully recovering data in the event of an emergency. For example, store your backup tapes in a safe place and don't abuse them. The following are tips to increase the reliability of your backups.

- Clean your tape drives regularly as recommended by your tape drive manufacturer.
- Backup tapes were not designed to last forever, so discard your backup tapes after the manufacturer's recommended cycles.

One of the most important aspects of your backup is testing your backup procedures and verifying your backups regularly to ensure that you're backing up your system accurately. You should practice your backup and restore procedures

---

regularly as described below.

1. Using a full backup from your normal backup set, restore your database files to a test server, and make sure the log files replay the way you expect them to.
2. Restore any incremental or differential backups built from the full backup to make sure they restore correctly.
3. Run some basic tests, for example, log on to a mailbox or access a public folder, to make sure the restored database is functioning properly.

It's useful to run the following test before you deploy Exchange Server in a production environment.

1. Back up your Exchange databases to tape.
2. Generate some load on the system so you get some activity in your log files. You can do this by running Loadsim or Mailstorm.
3. Simulate a crash.
4. Restore your database files from your tape backup and make sure the log files replay the way you expect them to.
5. Run some basic tests, for example, log on to a mailbox or access a public folder, to make sure the restored database is functioning properly.

---

## RECOVERING DATA

There are two types of failures that require the system to recover.

- A system crash causes a loss in the contents of memory. The system can typically repair this problem by performing a “soft recovery” when you restart the server.
- A hardware or software failure causes a corruption in the contents of a database. This is typically a serious problem that requires you to restore from backup.

### Soft Recovery

In many cases, Exchange is able to “silently” recover when a server crashes and the contents of the database buffer in memory is lost. It does this by performing a “soft recovery,” a process that runs automatically when you try to start the information store after a failure. Soft recovery uses the log files and database files on disk instead of tape backup to recover from a failure.

If your server crashes and the contents of memory are lost, the database file on disk is flagged as “inconsistent.” Before you can restart Exchange, the database must be made consistent again. Exchange uses soft recovery to simulate a clean shutdown by replaying pages from the log files on disk into the information store databases. Here’s how it works.

1. The database engine checks that the Edb.log file exists.
2. The database engine reads the checkpoint file to determine which log file to start replaying.

At the end of the operation, the database is effectively consistent again and the Exchange information store can start normally. If, however, soft recovery doesn’t work or if there’s a more serious problem with the system, you need to restore from backup.

### Restoring from Online Backup

Restoring from an online backup is a similar process to soft recovery. Exchange takes care of making sure that all the files are put in the right place and brings the database up to a consistent state.

1. When you restore data from tape backup, the restore process returns to disk all the files that were backed up.
2. The system attendant starts the information store.
3. The information store checks the RestoreInProgress key in the registry and determines that the database has been restored from an online backup.
4. The RestoreInProgress key tells the information store where to start replaying the transaction logs. It does not check for Edb.log.
5. The information store writes pages in the .Pat file into the database files, replays

---

the log files specified by the RestoreInProgress key, and plays through any other logs on disk.

If you need to restore a corrupt information store, a full recovery includes the following.

1. Fixing the problem.
2. Restoring the database.
3. Rolling forward all transaction log files.

Keep in mind that the most time-consuming aspect of doing a restore is not copying the database files back to the disk, but replaying the log files. In fact, depending on how often you perform full backups, it can take several hours to replay log files on large servers when doing a restore. This is because the database engine must run through every transaction that has occurred since the last backup. Transaction log replay speed also varies according to the type of transactions that must be replayed. For example, it takes the information store longer to replay log files with a lot of small delete operations or attachments. On average, it takes approximately 30 seconds to four minutes to replay each log file. You should test your log files to get a more accurate estimate of how long it takes to roll forward on your system.

---

### **Restoring Individual Items in a Mailbox**

Sometimes users delete messages that they later realize they really shouldn't have. Because Exchange handles backup and restore at the physical page layer not at the mailbox level, you can't easily restore individual messages in a mailbox from backup (some third-party backup programs are available that allow you to do a "brick backup;" however, they don't use the Exchange backup and restore APIs and typically don't perform as well as backups at the physical page layer.). That said, there IS a way users can recover messages they've deleted from their mailbox without having to resort to backups.

The Recover Deleted Items feature provided with Exchange 5.5 lets users retrieve messages from the Deleted Items folder in Outlook provided the administrator has enabled the feature on the server. The Recover Deleted Items feature is a handy way for users to get back that really important message they thought they had deleted the other day—without having to go to their administrator for help. However, keep in mind that if this feature is enabled, your server will require additional disk resources to store the deleted items.

In a future release, Exchange will be extended to allow applications to recover messages even after they've been permanently deleted from the system.

---

## MANAGING AND MAINTAINING YOUR DATABASES

You may have already heard that the information store in Exchange 5.5 is designed to run forever. As a result, you should rarely need to turn off your server to perform regular maintenance. Except for performing backups, Exchange takes care of all database maintenance automatically while the server is online. Most of these activities occur as background processes during regular information store maintenance. The information store typically runs online maintenance at night and performs important tasks such as Deleted Item cache processing, general clean up, and online defragmentation.

### Defragmenting Your Databases

One of the most important functions that the information store performs during regular online maintenance is reclaiming unused disk space by defragmenting the database. This feature has been fine-tuned since Exchange 5.0. In fact, Exchange 5.5 SP1 now includes a reporting tool that provides an estimate in the event log of how much free space is available in the information store after online defrag. This helps make the job of estimating how much disk space is needed a lot easier.

**Note:** The information store indicates in the event log when online defrag is started, interrupted, resumed, and completed. If you're backing up the information store or performing an offline defrag, check the event log to make sure you're not overlapping with online defrag. However, if online defrag is interrupted, the information store will resume the process at the next opportunity.

Online defragmentation does everything you need except shrink the size of the database files on disk. If you make major changes to your computer running Exchange (for example, if you move or delete a large number of mailboxes or remove a large number of newsgroups), you may consider performing an offline defrag by running `Eseutil /d`. However, in most cases, you should avoid running offline defrag because it's an expensive operation.

When offline defrag runs, it creates a new database file and then copies all the data in the old file to the new file. This can take a lot of time. On average, it takes about one hour to defragment 5 to 10 GB. What's more, you need enough free space for the offline defrag process to hold the new file. As a general rule, you should have 100 percent more free space than the amount you're defragmenting.

### Capacity Management

An important issue when planning your Exchange system is determining how much disk space you'll need for the information store. Without a conscientious approach to capacity management, the size of your information store databases will quickly get out of control.

We suggest that you set a maximum information store size and then manage the information store within those limits. You can get a good idea of how big your databases will grow by setting mailbox quotas and tracking the growth of the

---

information store over time. You should have enough free space to support the messaging needs of the users on the server, but mailbox storage limits should be set such that users don't consume excessive amounts of disk resources.

In addition to being proactive about planning disk space capacity, you should also have a plan for how you'll handle oversize information stores, should something go wrong with Plan A. This should include contingency plans for reclaiming disk space or adding extra disks to support your server's load. For example, you should have enough space on your system to allow you to run ESEUTIL and ISINTEG if needed. As a general rule, you should allow approximately 25-30 percent extra disk space for these utilities.

---

## PLANNING HARDWARE NEEDS

Because Exchange databases are all about quickly accessing large amounts of information, your primary concerns when selecting hardware for your databases are performance and reliability. You should always use high-quality components, because if you don't, you have a greater chance of losing data. And, if your hardware is too slow to support the load on your server, it can quickly become a bottleneck that slows down your entire system. For example, you should use RAID controllers with write caching with ECC memory protection and battery backup.

Note: You should be skeptical of OEM performance benchmarks since they typically test using RAID 0. You should use RAID 5 or 1 for database partitions to ensure the greatest reliability.

### Optimizing Database Access

For servers supporting large information stores (50 to 100 GB), it is especially important to follow these guidelines.

- Place transaction log files and database files on different disks.
- Dedicate a high performance spindle to the transaction logs.
- Use a dedicated partition for the databases. Experience shows that as servers get bigger, the database partition starts to use a lot of I/O. This is especially true for RAID 5 partitions because of the added overhead. As a result, it's a good idea to only put database files on the database partition.
- Put the MTA database and tracking logs on the system disk (if you don't have a spare spindle), not the database partition.
- Use NTFS instead of FAT for all Exchange database partitions, including partitions containing the information store databases and transaction log files.

The following sample disk configuration is recommended for typical large servers.

#### **Mirror set 1**

System disk. Includes binaries, swap file, MTA database.

#### **Mirror set 2**

Transaction files only.

#### **RAID 5 partition**

Exchange information store and directory databases only.

### Optimizing Backup and Restore Performance

Now let's talk about what is the real limiting factor when it comes to building large servers—how fast can you backup, and more importantly, how fast can you restore? To get the greatest performance out of your backup and restore process, you should use high performance backup software as well as the fastest backup hardware you can get your hands on. The following are just a few of the many

---

vendors who provide high performance backup software that works with Exchange.

- Cheyenne ARCserve
- Legato NetWorker
- Seagate BackupExec

Your backup hardware should consist of the following.

- Tape backup that supports streaming and striping.
- Fast-Wide SCSI. You should have 7 or 8 disks in an array if you're using more than one DLT in parallel.

If you outfit your system with quality components, you are probably wondering what kind of performance can you expect. If your system is properly configured, you should be able to reach the following speeds.

**Backup to a single DLT 35/70 tape drives**

Approximately 22-30 GB per hour (or 6.2 to 8.5 MB per second) with hardware compression. Note that performance will vary depending on the compression rate.

**Backup to RAID 5 array of 4 DLT 35/70 tape drives**

Approximately 40 GB per hour.

**Restore to RAID 5 disk partition**

Approximately 20 to 25 GB per hour with write-back caching. If write-back caching is disabled, it takes twice as long to restore.

---

## KEY MESSAGES

The following are the key messages that you should take away from this white paper.

- Do online (or copy) backups regularly, preferably every night.
- Monitor your backups, track the status of the database engine and the information store, and don't ignore error messages!
- Decide how big you'll let your information store grow, and manage it at that level.
- Trust Exchange 5.5 to manage the database internals (including database defragmentation).
- Test and practice your back up and restore procedures.